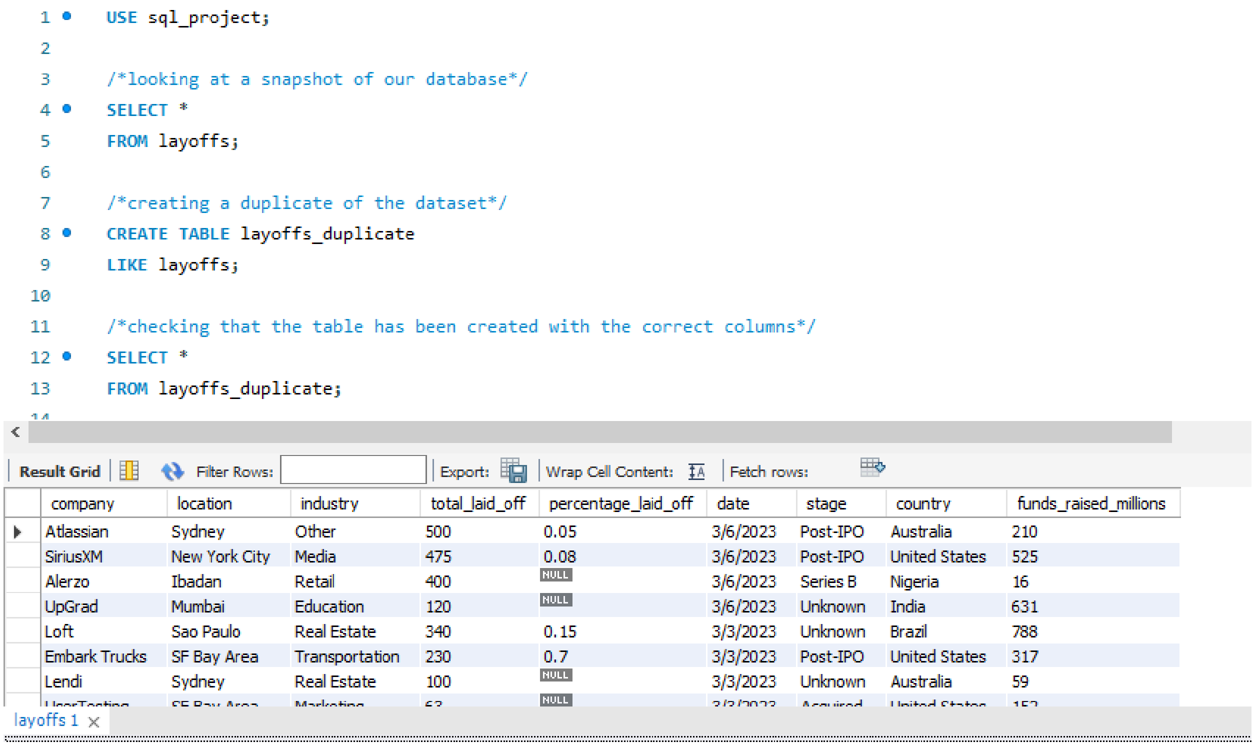
In this project, I will clean up a database and perform exploratory data analysis on it. I will do this by

- removing duplicates
- standardizing data
- addressing null values or blanks
- removing any rows and columns not needed
- conduct an EDA

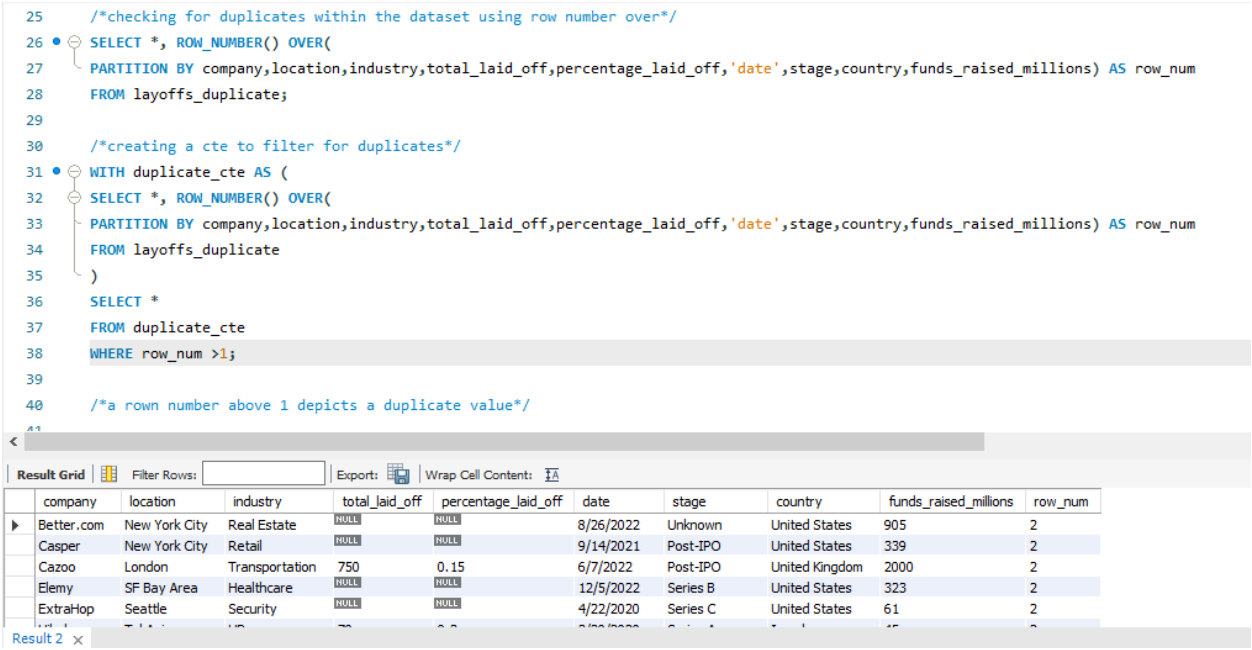
I start off by looking at a snapshot of our data. After first impressions, I create a duplicate of the database and store the original away to be referred to whenever needed.



```sql
USE sql_project;

/*looking at a snapshot of our database*/
SELECT *
FROM layoffs;

/*creating a duplicate of the dataset*/
CREATE TABLE layoffs_duplicate
LIKE layoffs;

/*checking that the table has been created with the correct columns*/
SELECT *
FROM layoffs_duplicate;
```

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
Atlassian	Sydney	Other	500	0.05	3/6/2023	Post-IPO	Australia	210
SiriusXM	New York City	Media	475	0.08	3/6/2023	Post-IPO	United States	525
Alerzo	Ibadan	Retail	400	NULL	3/6/2023	Series B	Nigeria	16
UpGrad	Mumbai	Education	120	NULL	3/6/2023	Unknown	India	631
Loft	Sao Paulo	Real Estate	340	0.15	3/3/2023	Unknown	Brazil	788
Embark Trucks	SF Bay Area	Transportation	230	0.7	3/3/2023	Post-IPO	United States	317
Lendi	Sydney	Real Estate	100	NULL	3/3/2023	Unknown	Australia	59
UserTesting	SF Bay Area	Marketing	63	NULL	3/3/2023	Acquired	United States	152

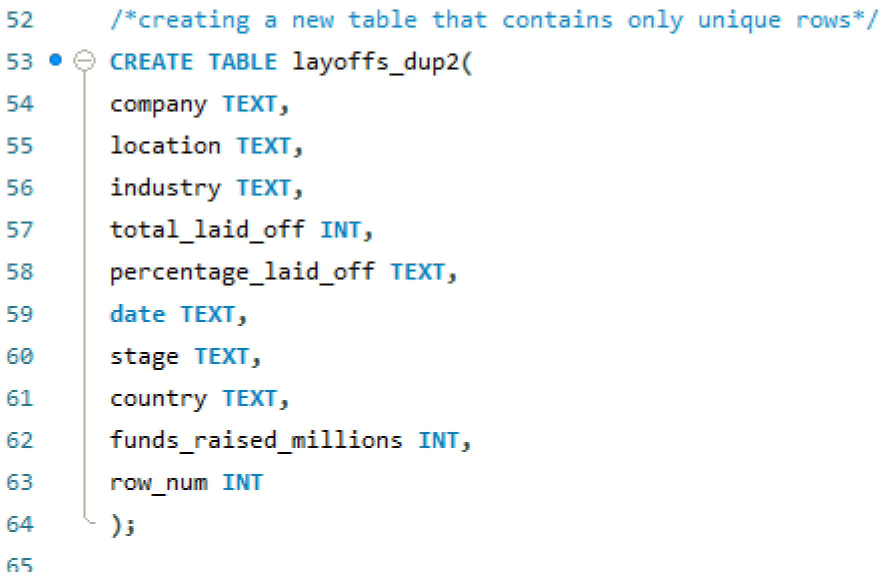
Using row_number() over(), I am able to suss out where a particular row has repeat datapoints in all columns meaning it is a duplicate.



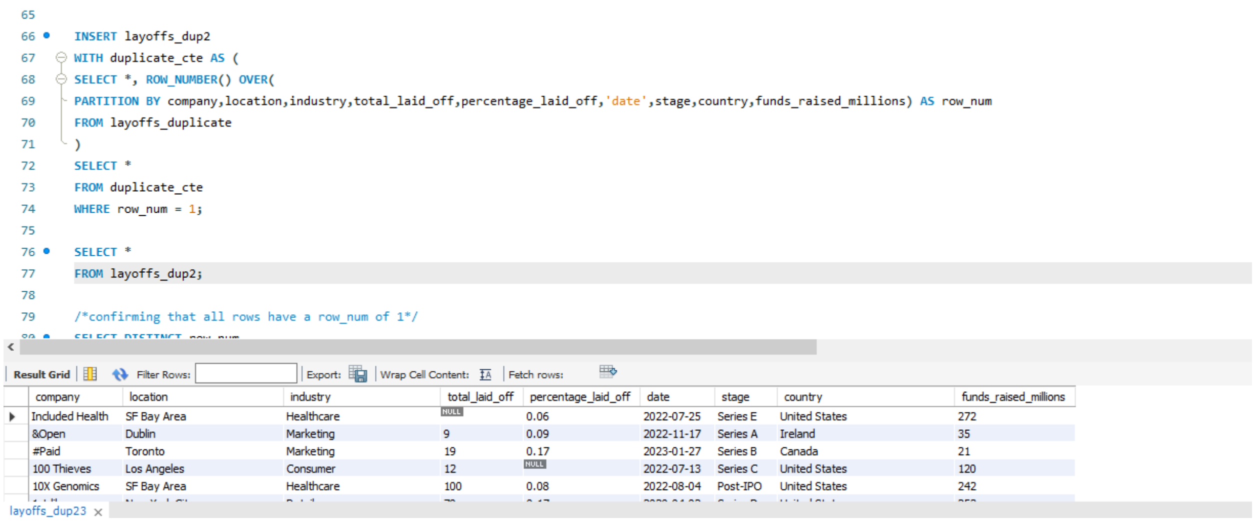
```sql
/*checking for duplicates within the dataset using row number over*/
SELECT *, ROW_NUMBER() OVER(
PARTITION BY company,location,industry,total_laid_off,percentage_laid_off,'date',stage,country,funds_raised_millions) AS row_num
FROM layoffs_duplicate;

/*creating a cte to filter for duplicates*/
WITH duplicate_cte AS (
SELECT *, ROW_NUMBER() OVER(
PARTITION BY company,location,industry,total_laid_off,percentage_laid_off,'date',stage,country,funds_raised_millions) AS row_num
FROM layoffs_duplicate
)
SELECT *
FROM duplicate_cte
WHERE row_num >1;

/*a rown number above 1 depicts a duplicate value*/
```

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_num
Better.com	New York City	Real Estate	NULL	NULL	8/26/2022	Unknown	United States	905	2
Casper	New York City	Retail	NULL	NULL	9/14/2021	Post-IPO	United States	339	2
Cazoo	London	Transportation	750	0.15	6/7/2022	Post-IPO	United Kingdom	2000	2
Elemy	SF Bay Area	Healthcare	NULL	NULL	12/5/2022	Series B	United States	323	2
ExtraHop	Seattle	Security	NULL	NULL	4/22/2020	Series C	United States	61	2

I create a second duplicate table and populate it with the unique data points, removing the duplicates.



```sql
/*creating a new table that contains only unique rows*/
CREATE TABLE layoffs_dup2(
company TEXT,
location TEXT,
industry TEXT,
total_laid_off INT,
percentage_laid_off TEXT,
date TEXT,
stage TEXT,
country TEXT,
funds_raised_millions INT,
row_num INT
);
```

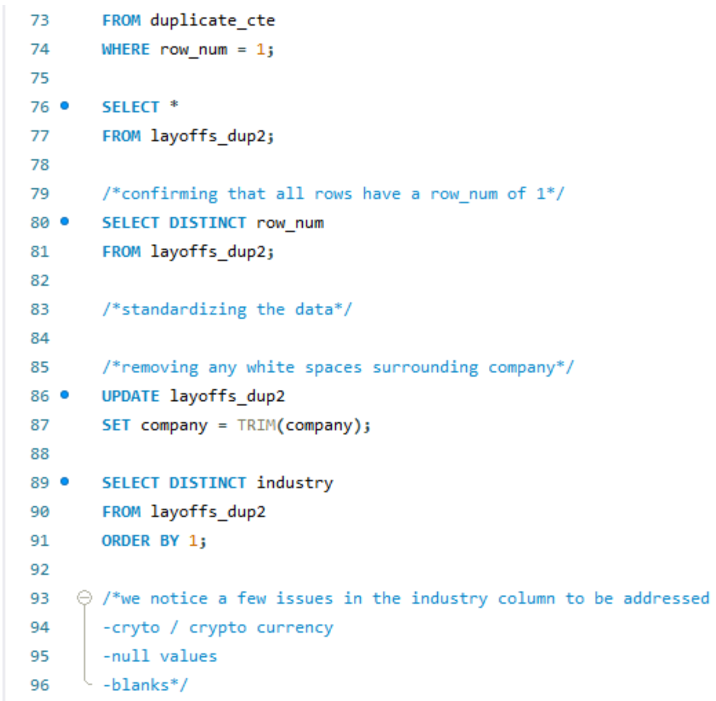


```
INSERT layoffs_dup2
WITH duplicate_cte AS (
SELECT *, ROW_NUMBER() OVER(
PARTITION BY company,location,industry,total_laid_off,percentage_laid_off,'date',stage,country,funds_raised_millions) AS row_num
FROM layoffs_duplicate
)
SELECT *
FROM duplicate_cte
WHERE row_num = 1;

SELECT *
FROM layoffs_dup2;

/*confirming that all rows have a row_num of 1*/
```

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
Included Health	SF Bay Area	Healthcare	NULL	0.06	2022-07-25	Series E	United States	272
&Open	Dublin	Marketing	9	0.09	2022-11-17	Series A	Ireland	35
#Paid	Toronto	Marketing	19	0.17	2023-01-27	Series B	Canada	21
100 Thieves	Los Angeles	Consumer	12	NULL	2022-07-13	Series C	United States	120
10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	Post-IPO	United States	242

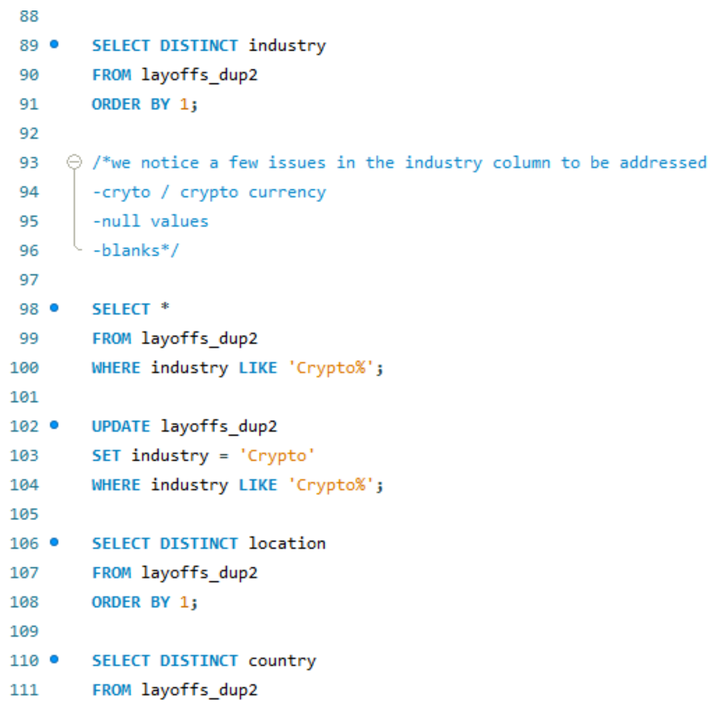
We proceed using the second duplicate table as our working dataset.

On to standardization, the column “company” contains words and spaces. The spaces sometimes come before the words or after, creating inconsistencies that may affect future analysis. I us the trim function to solve this issue.



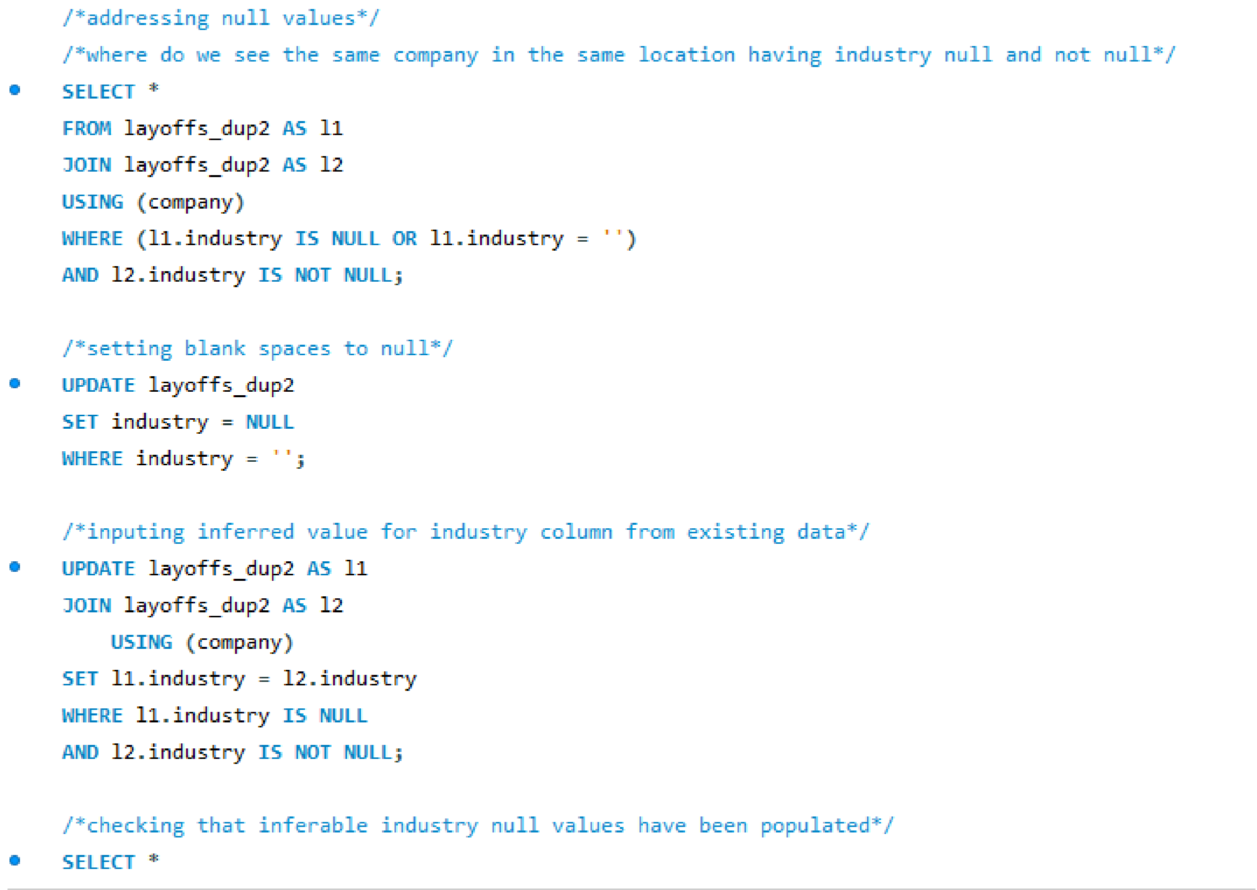
```
FROM duplicate_cte
WHERE row_num = 1;

SELECT *
FROM layoffs_dup2;

/*confirming that all rows have a row_num of 1*/
SELECT DISTINCT row_num
FROM layoffs_dup2;

/*standardizing the data*/

/*removing any white spaces surrounding company*/
UPDATE layoffs_dup2
SET company = TRIM(company);

SELECT DISTINCT industry
FROM layoffs_dup2
ORDER BY 1;

/*we notice a few issues in the industry column to be addressed
-cryto / crypto currency
-null values
-blanks*/
```

I move to the next column “industry”. The industry column has entries under both “Crypto” and “Cryto Currency” which I decide are the same thing. Since “Cryto” appears more frequently, I would be using that. Industry also has some null values and blanks.



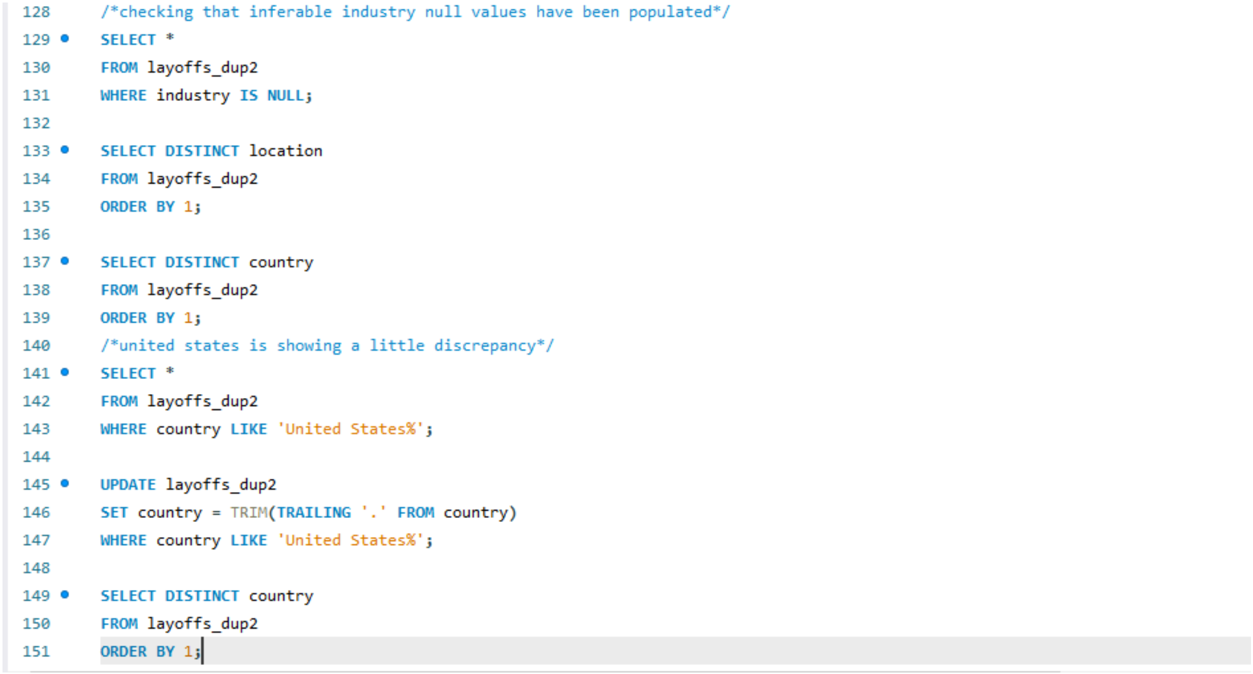
```sql

SELECT DISTINCT industry
FROM layoffs_dup2
ORDER BY 1;

/*we notice a few issues in the industry column to be addressed
-cryto / crypto currency
-null values
-blanks*/

SELECT *
FROM layoffs_dup2
WHERE industry LIKE 'Crypto%';

UPDATE layoffs_dup2
SET industry = 'Crypto'
WHERE industry LIKE 'Crypto%';

SELECT DISTINCT location
FROM layoffs_dup2
ORDER BY 1;

SELECT DISTINCT country
FROM layoffs_dup2
```

Next, I look at where a company in the same location appears more than once but has industry nulls or blanks in one of the entries. To make things easier, I will set the blanks to null values, the populate the nulls with industry values inferred from the existing data.



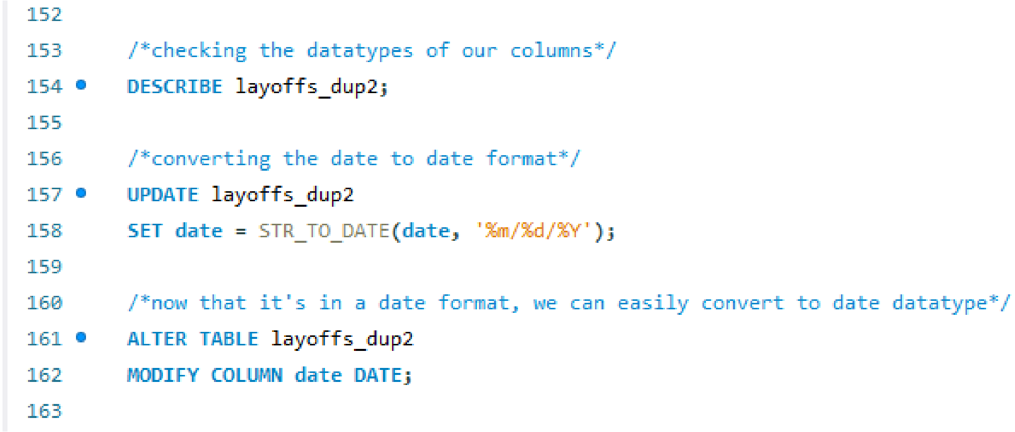
```
/*addressing null values*/
/*where do we see the same company in the same location having industry null and not null*/
SELECT *
FROM layoffs_dup2 AS l1
JOIN layoffs_dup2 AS l2
USING (company)
WHERE (l1.industry IS NULL OR l1.industry = '')
AND l2.industry IS NOT NULL;

/*setting blank spaces to null*/
UPDATE layoffs_dup2
SET industry = NULL
WHERE industry = '';

/*inputing inferred value for industry column from existing data*/
UPDATE layoffs_dup2 AS l1
JOIN layoffs_dup2 AS l2
    USING (company)
SET l1.industry = l2.industry
WHERE l1.industry IS NULL
AND l2.industry IS NOT NULL;

/*checking that inferable industry null values have been populated*/
SELECT *
```

After confirming that inferable entries have been made, I move to the next column “country”. There are 2 different “United States” because one has a period at the end. We need to remove said period so that the system recognizes them as one.



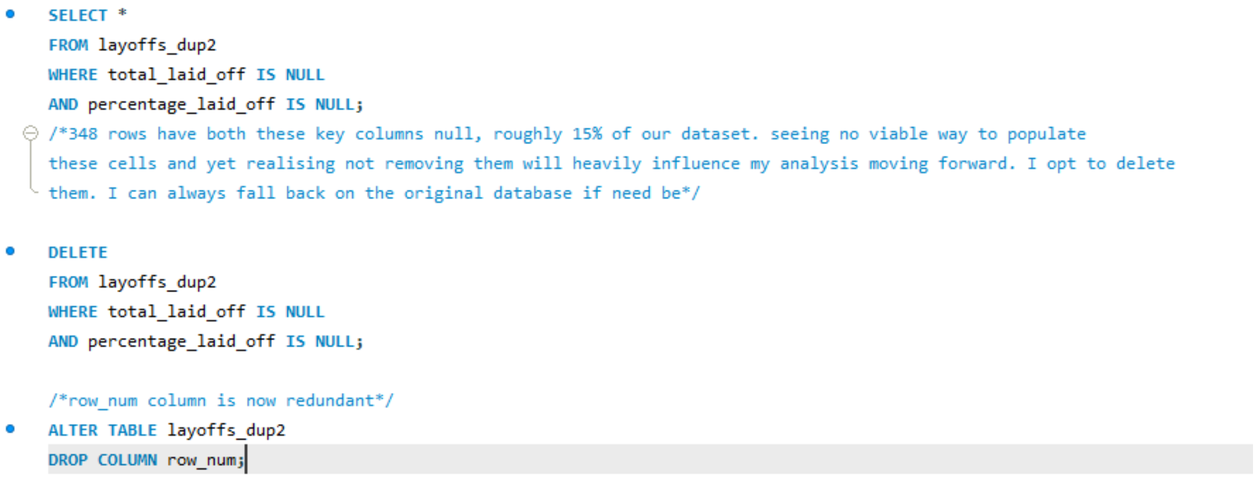
```sql
/*checking that inferable industry null values have been populated*/
SELECT *
FROM layoffs_dup2
WHERE industry IS NULL;

SELECT DISTINCT location
FROM layoffs_dup2
ORDER BY 1;

SELECT DISTINCT country
FROM layoffs_dup2
ORDER BY 1;
/*united states is showing a little discrepancy*/
SELECT *
FROM layoffs_dup2
WHERE country LIKE 'United States%';

UPDATE layoffs_dup2
SET country = TRIM(TRAILING '.' FROM country)
WHERE country LIKE 'United States%';

SELECT DISTINCT country
FROM layoffs_dup2
ORDER BY 1;
```

Checking the data types of the columns, I discover that date column is a text data type. I convert it to the more appropriate ‘date’ data type.



```sql

/*checking the datatypes of our columns*/
DESCRIBE layoffs_dup2;

/*converting the date to date format*/
UPDATE layoffs_dup2
SET date = STR_TO_DATE(date, '%m/%d/%Y');

/*now that it's in a date format, we can easily convert to date datatype*/
ALTER TABLE layoffs_dup2
MODIFY COLUMN date DATE;

```

My analysis will focus on the numeric aspect of the dataset, having null values the numeric columns will be problematic.

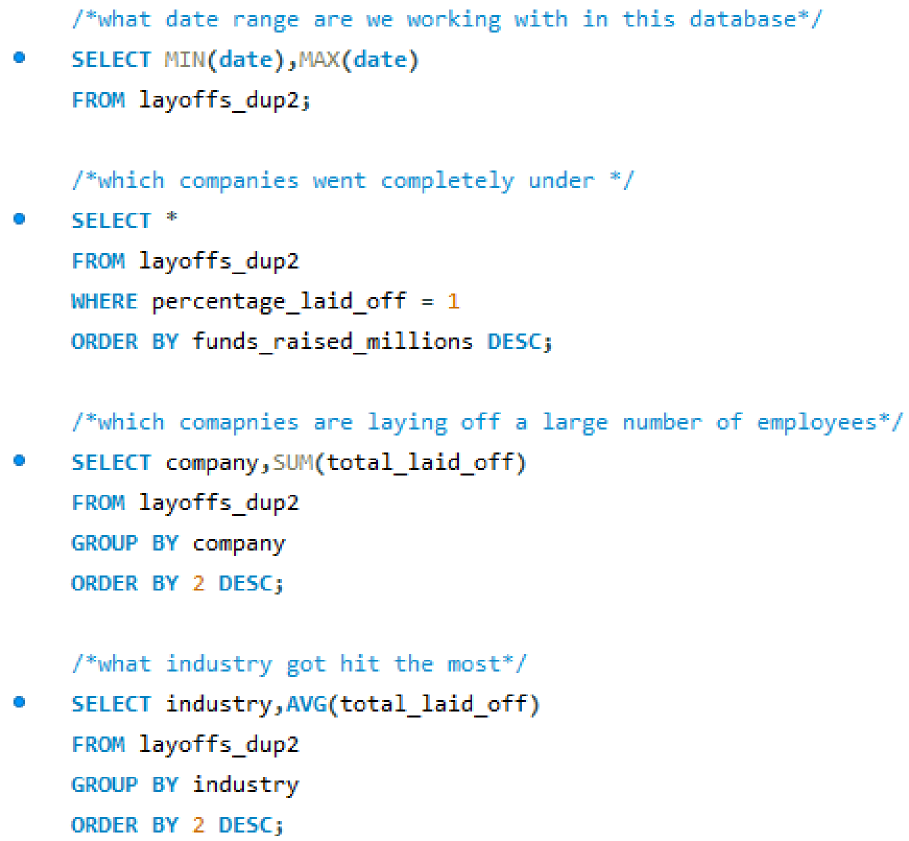


```
SELECT *
FROM layoffs_dup2
WHERE total_laid_off IS NULL
AND percentage_laid_off IS NULL;
/*348 rows have both these key columns null, roughly 15% of our dataset. seeing no viable way to populate
these cells and yet realising not removing them will heavily influence my analysis moving forward. I opt to delete
them. I can always fall back on the original database if need be*/

DELETE
FROM layoffs_dup2
WHERE total_laid_off IS NULL
AND percentage_laid_off IS NULL;

/*row_num column is now redundant*/
ALTER TABLE layoffs_dup2
DROP COLUMN row_num;
```

Checking for null values in our 2 numeric columns, I find that 348 rows have both numeric columns null, this accounts for about 15% of our original data. Unfortunately, I can not infer these values from the existing data, nor can I confidently input a calculated estimate. I decided that leaving these rows in will affect my analysis so opt to remove them. Also, the row_num column I used to find the duplicates is no longer needed so I remove that as well.

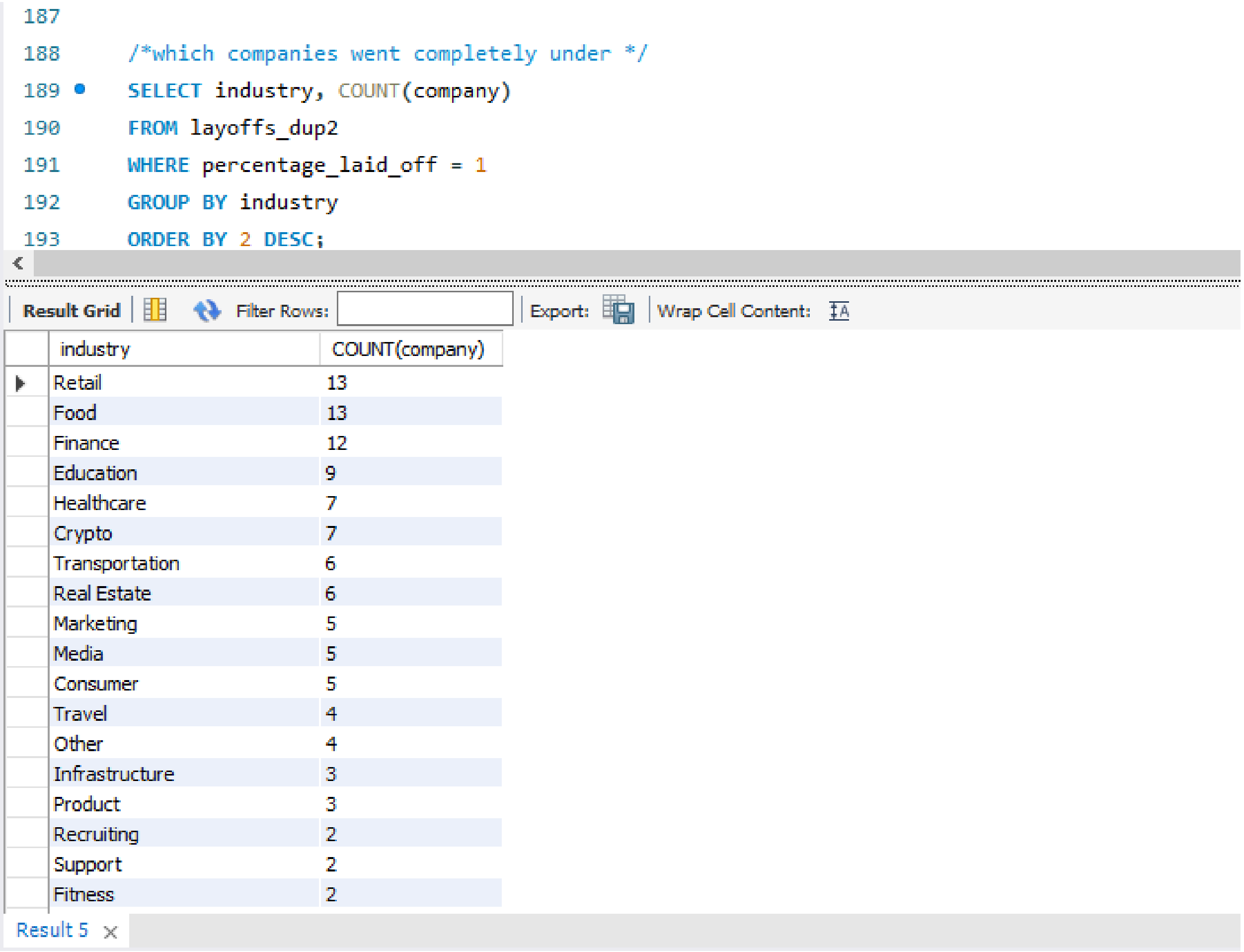
Feeling content with the cleaning for now, I move on to my exploratory data analysis. Things I am curious about the data set include:

- What is the timeframe of the data set ?
- Which companies went completely under during the period ?
- Which companies are laying off the most employees ?
- Which industry laid off the most on average ?
- Which countries laid off the most ?
- How did layoffs progress over the period of interest ?
- How the stage a company is in affected layoffs.



```
/*what date range are we working with in this database*/
SELECT MIN(date),MAX(date)
FROM layoffs_dup2;

/*which companies went completely under */
SELECT *
FROM layoffs_dup2
WHERE percentage_laid_off = 1
ORDER BY funds_raised_millions DESC;

/*which comapnies are laying off a large number of employees*/
SELECT company,SUM(total_laid_off)
FROM layoffs_dup2
GROUP BY company
ORDER BY 2 DESC;

/*what industry got hit the most*/
SELECT industry,AVG(total_laid_off)
FROM layoffs_dup2
GROUP BY industry
ORDER BY 2 DESC;
```

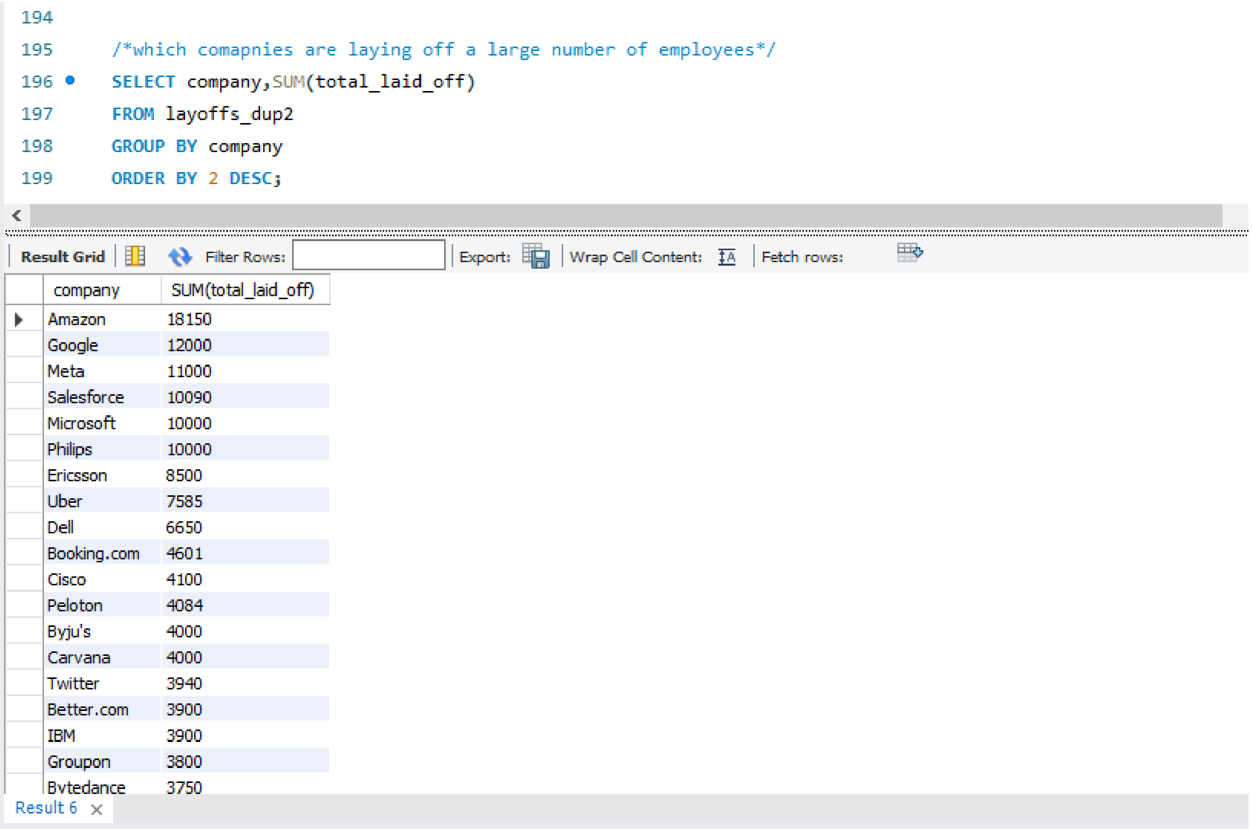
The data set is from 11th March 2020 to to 6th March 2023, starting at the peak of the world wide outbreak of the corona virus and afterwards.



```sql

/*which companies went completely under */
SELECT industry, COUNT(company)
FROM layoffs_dup2
WHERE percentage_laid_off = 1
GROUP BY industry
ORDER BY 2 DESC;
```

industry	COUNT(company)
Retail	13
Food	13
Finance	12
Education	9
Healthcare	7
Crypto	7
Transportation	6
Real Estate	6
Marketing	5
Media	5
Consumer	5
Travel	4
Other	4
Infrastructure	3
Product	3
Recruiting	2
Support	2
Fitness	2

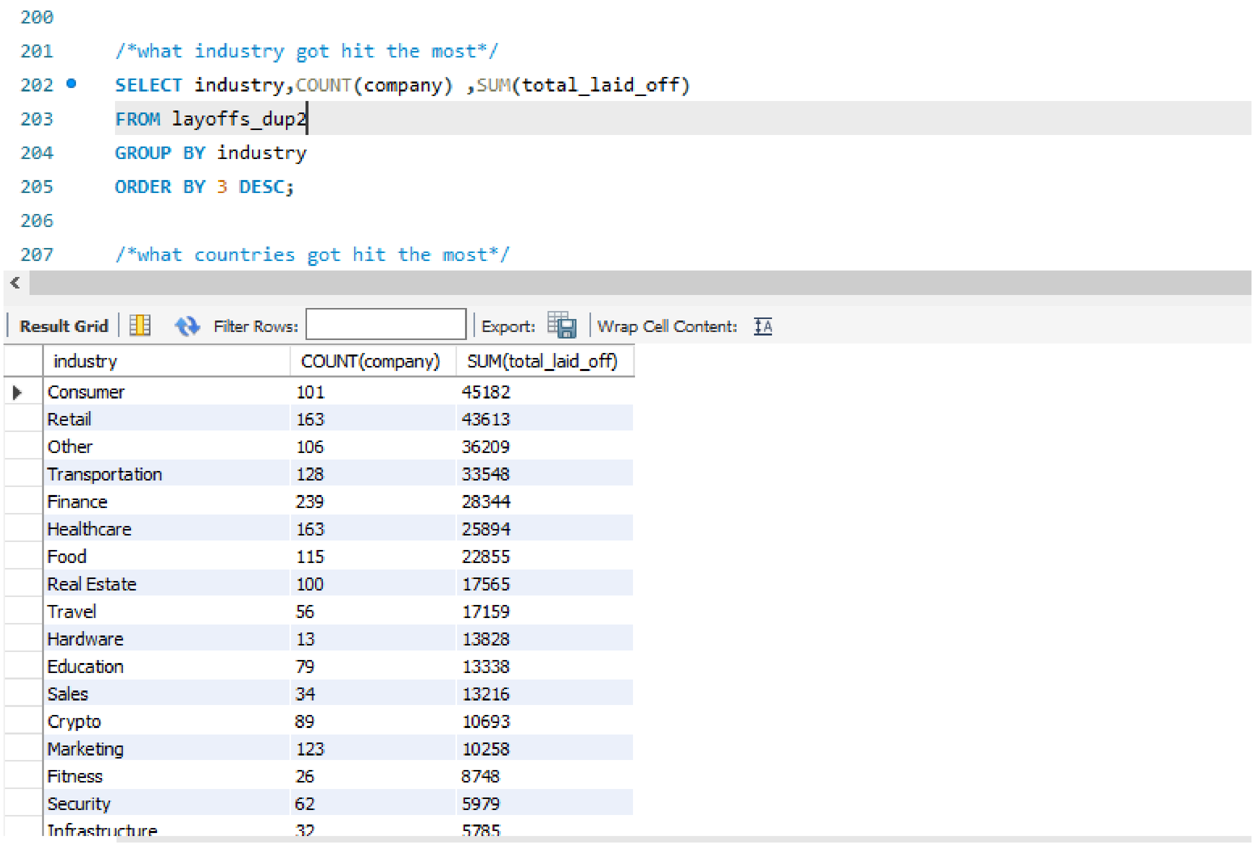
At the top of the list of industries we have retail, food and finance.



```
/*which comapnies are laying off a large number of employees*/
SELECT company,SUM(total_laid_off)
FROM layoffs_dup2
GROUP BY company
ORDER BY 2 DESC;
```

company	SUM(total_laid_off)
Amazon	18150
Google	12000
Meta	11000
Salesforce	10090
Microsoft	10000
Philips	10000
Ericsson	8500
Uber	7585
Dell	6650
Booking.com	4601
Cisco	4100
Peloton	4084
Byju's	4000
Carvana	4000
Twitter	3940
Better.com	3900
IBM	3900
Groupon	3800
Bytedance	3750

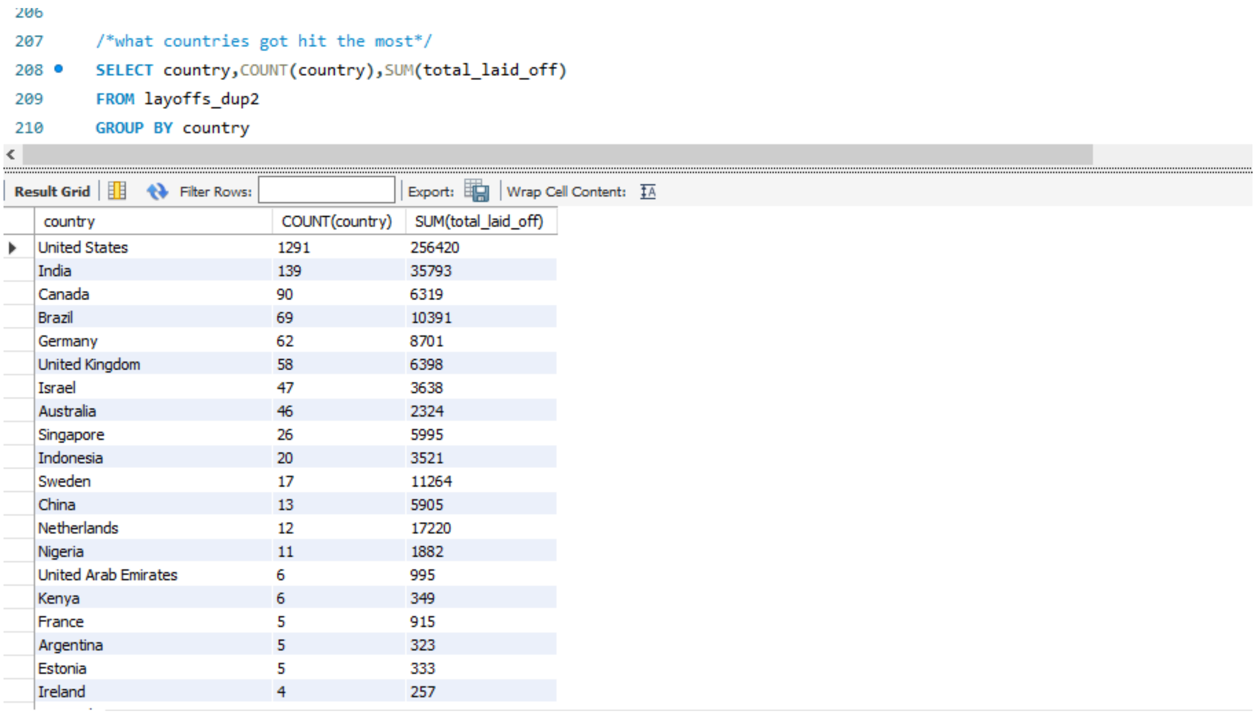
Large companies like Amazon, Google and Meta laid off the most employees during the period. However, these companies also have a disproportionately large number of employees in general.



```

/*what industry got hit the most*/
SELECT industry,COUNT(company) ,SUM(total_laid_off)
FROM layoffs_dup2
GROUP BY industry
ORDER BY 3 DESC;

/*what countries got hit the most*/
```

industry	COUNT(company)	SUM(total_laid_off)
Consumer	101	45182
Retail	163	43613
Other	106	36209
Transportation	128	33548
Finance	239	28344
Healthcare	163	25894
Food	115	22855
Real Estate	100	17565
Travel	56	17159
Hardware	13	13828
Education	79	13338
Sales	34	13216
Crypto	89	10693
Marketing	123	10258
Fitness	26	8748
Security	62	5979
Infrastructure	32	5785

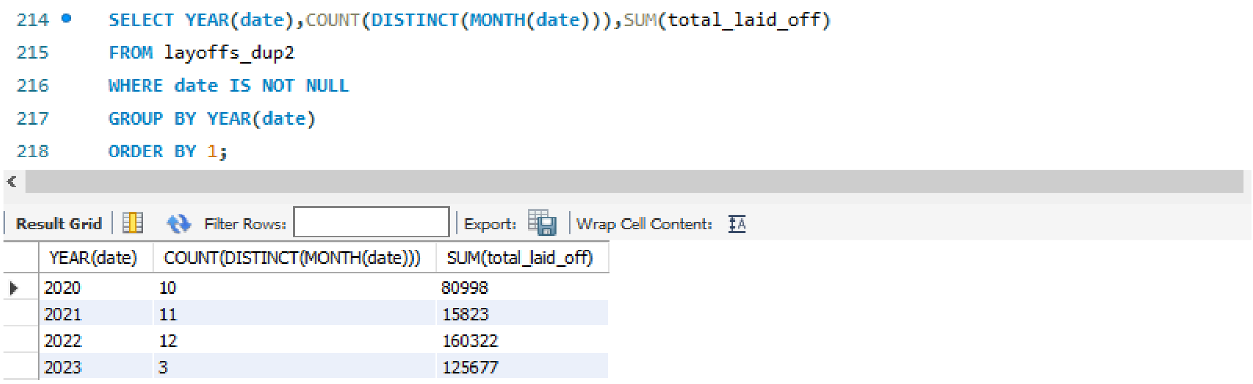
Consumer and retail are leading industries in layoffs.



```

/*what countries got hit the most*/
SELECT country,COUNT(country),SUM(total_laid_off)
FROM layoffs_dup2
GROUP BY country
```

country	COUNT(country)	SUM(total_laid_off)
United States	1291	256420
India	139	35793
Canada	90	6319
Brazil	69	10391
Germany	62	8701
United Kingdom	58	6398
Israel	47	3638
Australia	46	2324
Singapore	26	5995
Indonesia	20	3521
Sweden	17	11264
China	13	5905
Netherlands	12	17220
Nigeria	11	1882
United Arab Emirates	6	995
Kenya	6	349
France	5	915
Argentina	5	323
Estonia	5	333
Ireland	4	257

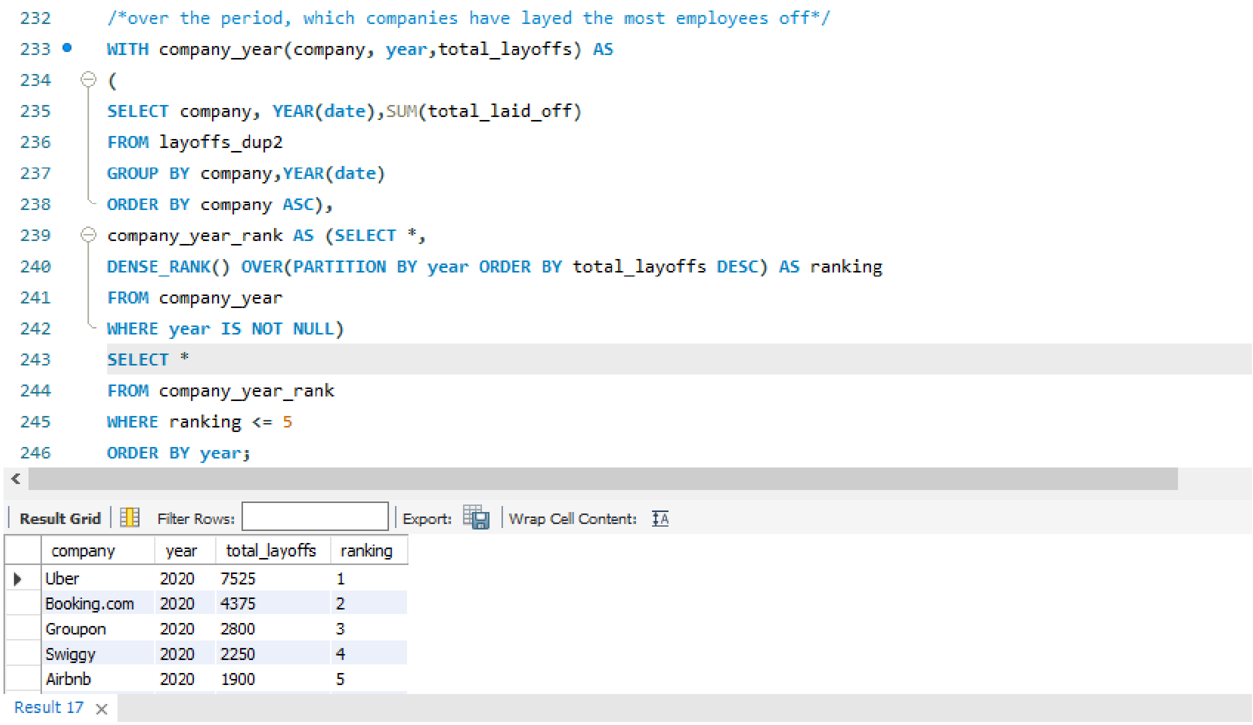
USA contributing the most companies to the database naturally also has the most number of layoffs.



```
SELECT YEAR(date),COUNT(DISTINCT(MONTH(date))),SUM(total_laid_off)
FROM layoffs_dup2
WHERE date IS NOT NULL
GROUP BY YEAR(date)
ORDER BY 1;
```

YEAR(date)	COUNT(DISTINCT(MONTH(date)))	SUM(total_laid_off)
2020	10	80998
2021	11	15823
2022	12	160322
2023	3	125677

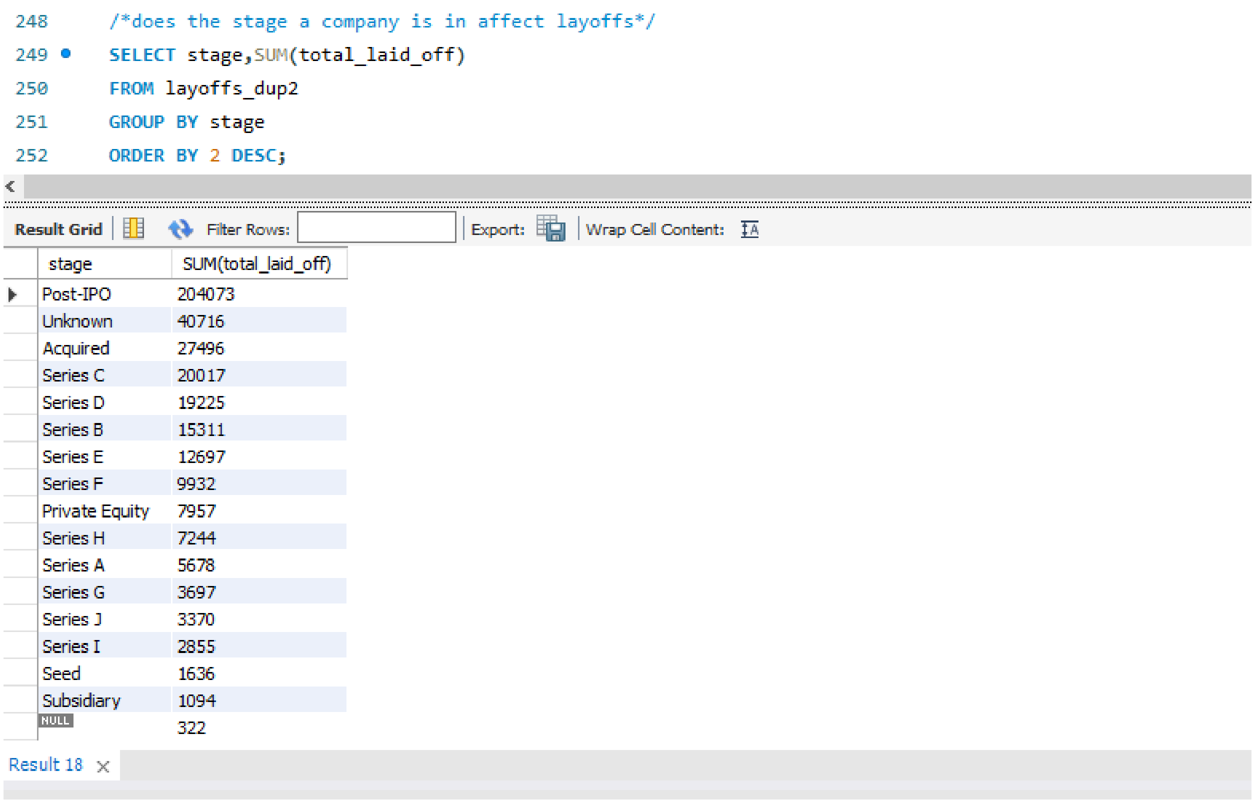
The layoffs were lowest in 2020, but subsequent years saw higher numbers in the aftermath of the pandemic. Though 2023 shows a reduction in layoffs, only 3 months out of the year are accounted for.



```
/*over the period, which companies have layed the most employees off*/
WITH company_year(company, year,total_layoffs) AS
(
SELECT company, YEAR(date),SUM(total_laid_off)
FROM layoffs_dup2
GROUP BY company,YEAR(date)
ORDER BY company ASC),
company_year_rank AS (SELECT *,
DENSE_RANK() OVER(PARTITION BY year ORDER BY total_layoffs DESC) AS ranking
FROM company_year
WHERE year IS NOT NULL)
SELECT *
FROM company_year_rank
WHERE ranking <= 5
ORDER BY year;
```

company	year	total_layoffs	ranking
Uber	2020	7525	1
Booking.com	2020	4375	2
Groupon	2020	2800	3
Swiggy	2020	2250	4
Airbnb	2020	1900	5

The top 5 companies laying off employees over the period was quite volatile.



```sql
/*does the stage a company is in affect layoffs*/
SELECT stage,SUM(total_laid_off)
FROM layoffs_dup2
GROUP BY stage
ORDER BY 2 DESC;
```

stage	SUM(total_laid_off)
Post-IPO	204073
Unknown	40716
Acquired	27496
Series C	20017
Series D	19225
Series B	15311
Series E	12697
Series F	9932
Private Equity	7957
Series H	7244
Series A	5678
Series G	3697
Series J	3370
Series I	2855
Seed	1636
Subsidiary	1094
NULL	322

Finally, companies in Post-IPO stage laid off the most during the period.